Propuesta para documentar trabajos finales utilizando metodologías ágiles

Marcelo Uva, Marcela Daniele, Fabio Zorzan, Mariana Frutos y Ariel Arsaute

Departamento de Computación, FCEFQyN, Universidad Nacional de Río Cuarto, Río Cuarto, Argentina.
Email: {uva,marcela,fzorzan,mfrutos,aarsaute}@dc.exa.unrc.edu.ar

Resumen

La aplicación de metodologías ágiles en proyectos de desarrollo de software ha sido muy significativa y creciente en los últimos años. En el informe World Quality Report (WQR) presentado por la consultora Sogeti, Capgemini y HP en 2013 sobre inversiones de las empresas en Tecnologías de la Información se indica que el 83% de las empresas usan metodologías ágiles para el desarrollo de sus aplicaciones debido a la versatilidad ante los cambios. Un modelo de desarrollo ágil, define un proceso incremental, donde la comunicación entre clientes y desarrolladores es constante y colaborativa. En los ámbitos académicos, en un principio, fue resistida la aplicación de estas nuevas metodologías. La informalidad y la aparente escasez en la documentación fueron los puntos más cuestionados. En este trabajo se presenta una plantilla para documentar proyectos finales de carreras de computación, siguiendo un desarrollo ágil. La misma está destinada a alumnos avanzados de carreras de informática que estén iniciando su trabajo final de carrera. Se pretende realizar aportes para establecer un nexo entre los postulados fundacionales de las metodologías ágiles y la realidad que enfrenta un estudiante sin experiencia al momento de iniciar su trabajo de fin de carrera.

Palabras clave: Proyectos finales, Metodologías Ágiles, SCRUM, Documentación.

1 Introducción

En 1979 Tom DeMarco [1] definió al desarrollo de software como la generación de un producto de ingeniería, que requiere de planificación y de la construcción de modelos de diseño. Tal como resulta en la ingeniería clásica, los modelos proveen un medio de comunicación entre todos los participantes, clientes, usuarios y desarrolladores. Una metodología de desarrollo de software, establece una secuencia de actividades a seguir para completar el ciclo de vida de desarrollo de un sistema [2]. Como en cualquier tipo de proyecto, es necesaria una adecuada gestión de recursos para lograr el éxito. La gerencia de un proyecto de software plantea la obtención de un producto de calidad, planificando, organizando, supervisando y controlando su evolución durante todo el ciclo de vida [3]. Para ello se debe gestionar adecuadamente al conjunto: personal, proceso y producto. También es de vital importancia la selección de los indicadores adecuados para la medición del proyecto y la obtención de estimaciones confiables en cuanto a costos, duración y recursos en general. El crecimiento del uso de metodologías ágiles dentro de los ámbitos de producción de software ha sido muy significativa en estos últimos años.
Un modelo de desarrollo ágil, define un proceso incremental, que generalmente consiste en pequeñas entregas en ciclos cortos. La comunicación entre clientes y desarrolladores es constante y colaborativa. El "Manifiesto Ágil" [4] es un documento creado en el año 2001 por un pequeño grupo de expertos en el desarrollo de software encabezados por Kent Beck. En este documento se presentan los principios emergentes de una nueva manera de gestionar proyectos. El Manifiesto, no es, en sí mismo, una metodología, sino una reflexión de alto nivel que debería seguir la gestión de proyectos de software. Los cuatro principios fundamentales del manifiesto pueden resumirse en las siguientes ideas: los individuos e interacciones por encima de procesos y herramientas, el software funcionando por encima de documentación extensiva, colaboración con el cliente por encima negociación contractual y respuesta ante el cambio por encima de seguir un

plan. Existe una creencia popular que dice que cuando se utilizan metodologías ágiles no es necesario producir ningún tipo de documentación, sólo es cuestión de sentarse a programar.
En los ámbitos académicos, fue resistida en un principio la aplicación de estas nuevas metodologías. La informalidad y la aparente escasez en la documentación producida, fueron uno de los puntos más cuestionados.
En este trabajo se presenta una plantilla para documentar proyectos que siguen una metodología de desarrollo ágil. La propuesta está destinada principalmente a alumnos avanzados de carreras de informática que están iniciando su trabajo de fin de carrera. Se busca realizar aportes para lograr un nexo entre los postulados fundacionales de las metodologías ágiles y la realidad de un estudiante, sin experiencia generalmente, al momento de comenzar el desarrollo de su trabajo final de carrera. La plantilla está basada en la metodología SCRUM [5] pero es posible su extensión para cualquier metodología ágil del estilo. La propuesta establece una serie de documentos que deberían producirse como resultado de la aplicación de una metodología como SCRUM.
Este trabajo está organizado de la siguiente manera: en la sección 2 se realiza una fundamentación de la propuesta junto con una reseña de las tareas de investigación vinculadas a la misma. En la sección 3 se presentan diferentes enfoques vinculados a metodologías tradicionales y ágiles para el desarrollo de sistemas. En la sección 4, la propuesta para documentar proyectos finales utilizando metodologías ágiles. La implementación de la propuesta en la sección 5 y finalmente en la sección 6 los resultados obtenidos junto con las conclusiones.

2 Fundamentos

Desde hace más de 10 años, los autores de este trabajo y docentes de las asignaturas correspondiente al último año de las carreras de Analista en Ciencias de la Computación de la Universidad Nacional de Río Cuarto (U.N.R.C.), analizan y estudian diferentes problemáticas relacionadas a la enseñanza y el aprendizaje del análisis, diseño y desarrollo de sistemas de software. Se han propuesto diferentes alternativas para mejorar y favorecer el dictado de algunos temas específicos en las asignaturas involucradas. En el año 2004 [6], los autores de este trabajo, llevaron adelante el proyecto innovador "Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso". En este proyecto se presentó la necesidad de plantear soluciones genéricas a problemas similares. El mencionado proyecto cubría la primera etapa en el proceso de desarrollo de un sistema, la captura de requerimientos. Se propuso también la definición de plantillas genéricas para la descripción de casos de uso en los problemas de: inserción, eliminación, modificación y búsqueda de un elemento. En el año 2005 en [7], se propuso continuar con la definición de plantillas genéricas para cubrir las siguientes etapas del proceso de desarrollo. Se trabajó en un nuevo proyecto llamado "Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas Genéricas para Análisis y Diseño". Con la definición de plantillas genéricas para la etapa de Análisis y Diseño, se completaron las tres etapas fundamentales. En el año 2006 en [8], se continuó apostando al mejoramiento de las prácticas docentes en la enseñanza de importantes temas vinculados a la ingeniería de software. Se propuso en ese entonces el proyecto "La enseñanza de gestión de proyectos de software y la aplicación de herramientas que favorezcan su automatización". Esta vez poniendo el foco en mejorar las prácticas vinculadas con la gestión de proyectos de software, con el convencimiento de que la utilización de herramientas iban a permitir una mejora en la gestión de proyectos. El objetivo fue ubicar al estudiante en roles gerenciales en el proceso de desarrollo ingenieril del software, logrando que éste aplique los conceptos adquiridos mediante la utilización de herramientas. Desde el año 2010, en [9][10][11], el grupo de trabajo se ha abocado a detectar y analizar las causas de los retrasos en la realización del proyecto final de la carrera Analista en Computación. Una de las cuestiones analizadas en estos trabajos ha sido la elección de las metodologías de desarrollo. En muchos casos se detectó una elección inadecuada de la metodología de acuerdo a la naturaleza del problema a resolver. Otro factor analizado ha sido la complejidad de los trabajos abordados, en ocasiones la complejidad superaba ampliamente el requerido por un trabajo de analista (estimado en seis meses aproximadamente) y que se adecuaban más al trabajo de una licenciatura (un año aproximadamente). El retraso de los estudiantes de carreras de computación para finalizar su trabajo final en los tiempos previstos, es una problemática observable por los docentes encargados de dirigir estos proyectos. En algunos casos hasta abandonan sus estudios a muy poco tiempo de su concreción. Este problema ha desvelando tanto a los académicos como a organismos estatales. Desde hace algunos años el Estado Nacional viene

analizando la oferta respecto de la demanda de profesionales de la industria del software. Se han implementado políticas que tienden a favorecer el incremento en los ingresantes, disminuir la deserción, analizar el desgranamiento y estimular a los estudiantes de los últimos años para que finalicen sus estudios. En pos de generar los medios que permitan favorecer a que los proyectos se ejecuten y finalicen en los tiempos planificados, y sabiendo que las metodologías tradicionales, en muchos casos, no se adaptan a las necesidades o expectativas que tienen los usuarios, se propone una plantilla tendiente a guiar el proceso de documentación vinculado a un proceso de desarrollo ágil. Esta plantilla establece que tipos de documentos serán necesarios en cada etapa, estableciendo un índice de documentos.

3 Metodologías Tradicionales y Metodologías Ágiles

Las metodologías tradicionales están centradas especialmente en el control del proceso de desarrollo, mediante una rigurosa definición de roles, actividades, artefactos, notaciones para el modelado y documentación detallada. Siendo estas muy efectivas y necesarias en proyectos de gran tamaño y complejidad. Por su parte, las metodologías ágiles, dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas produciendo documentos centrados en lo esencial. Adaptándose con mayor facilidad a otros tipos de proyectos. En el marco de los cursos de *Análisis y Diseño de Sistemas* e *Ingeniería de Software*, ambos del tercer año de las carreras Analista en Computación y Licenciatura en Ciencias de la Computación de la U.N.R.C, se presentan diferentes metodologías para el desarrollo de software. Algunos de los enfoques estudiados son: el ciclo de vida lineal o secuencial, el método del análisis estructurado de Jourdon [12] , desarrollo basado en componentes, diseño por contratos [13] de Bertrand Meyer, y el Proceso Unificado [14]. En particular se realiza una instanciación de esta metodología que propone un desarrollo basado en casos de uso, iterativo e incremental y centrado en la arquitectura. Se realizan trabajos prácticos donde se plantean situaciones problemáticas y se construyen los artefactos propuestos por la metodología. Por otro lado, se estudian metodologías ágiles de desarrollo de software analizando y evaluando alternativas. SCRUM [5], Crystal Methodologies, ASD (Adaptive Software Development, FDD (Feature-Driven Development), DSDM (Dynamic Systems Development Method). El curso de *Análisis y Diseño de Sistemas* posee un taller de desarrollo en el cual se realiza una instanciación de la metodología SCRUM. Se realizan comparaciones, y análisis de las particularidades, debilidades y fortalezas de las metodologías ágiles y las metodologías tradicionales.

3.1 SCRUM

Scrum es una metodología ágil que permite trabajar en ambientes muy cambiantes, con permanentes replanteamientos. Por otro lado permite reducir el tiempo de producción y de comercialización del producto obtenido, aporta un gran beneficio o valor agregado al cliente, optimiza esfuerzo/tiempo en la construcción de artefactos. Facilita también la comunicación entre todos los integrantes del proyecto. La documentación producida dentro de un proyecto SCRUM es relativa al usuario, dueño, producto y equipo. SCRUM es más bien un marco de trabajo que reposa sobre la premisa de que el equipo de desarrollo conocerá la mejor manera de resolver el problema que se le presenta. La reunión de planificación de cada grupo de requerimientos a producir se describe en términos del resultado deseado, en lugar de un conjunto de criterios de ingreso, definiciones y tareas. SCRUM se basa en una autoorganización, con un equipo multifuncional y sin líder (dentro del equipo). El equipo es apoyado por dos individuos quienes ocupan los roles de Scrum Master y Product Owner. El Scrum Master es una especie de entrenador para el equipo, su función es ayudar a los miembros del mismo a utilizar el marco que ofrece la metodología para conseguir un nivel alto de productividad. Mientras que el Product Owner representa los negocios, clientes o usuarios. Éste guía al equipo hacia la construcción del producto adecuado. Los proyectos SCRUM avanzan en orden a la definición de los sprints, que son las iteraciones que poseen una duración de entre dos y cuatro semanas. En el inicio de cada sprint, los miembros del equipo se comprometen a producir una cantidad de características que enumeran en el artefacto conocido como el Product Backlog del proyecto. Al final de cada sprint, cada funcionalidad (conocida como *historia*) debe estar codificada, probada e integrada a una versión demo del sprint anterior. Luego se realiza una revisión, y por último se demuestra la nueva funcionalidad frente al Pro-

duct Owner y las otras partes interesadas que proporcionen información requerida para el siguiente sprint. Las iteraciones han de continuar hasta obtener el producto deseado. Como se puede observar de lo expuesto, SCRUM establece una forma de trabajo en donde todo el equipo se autoregula y en donde no hay de antemano una documentación establecida. Cada equipo utilizará los elementos que le sean necesarios para poder llevar adelante el grupo de *historias* con el cual se ha comprometido. Algunos podrán utilizar diagramas UML [15] otros utilizarán diagramas de flujos de datos, etc. En la figura 1 se esquematiza todo el proceso.

4 Plantilla para documentar proyectos finales en SCRUM

Este trabajo esta orientado a facilitar la tarea de documentar proyectos finales llevados a cabo por alumnos avanzados siguiendo una metodología ágil. Para este trabajo se ha tomado SCRUM como base, pero puede ser extendida a cualquier otra metodología similar. Inicialmente, se propone contextualizar el ámbito donde se desarrollará el proyecto de software, describiendo brevemente el propósito del mismo, los objetivos a alcanzar, los actores involucrados y el valor agregado que brindará el producto resultante al cliente final.
Se indica también la construcción del documento con la especificación de requerimientos de software SRS (Specification Requeriments Sofware). En éste se describirán las funcionalidades que contendrá el producto de software, justificando cada una de ellas. Además, será necesario identificar los objetos más importantes involucrados su relación.

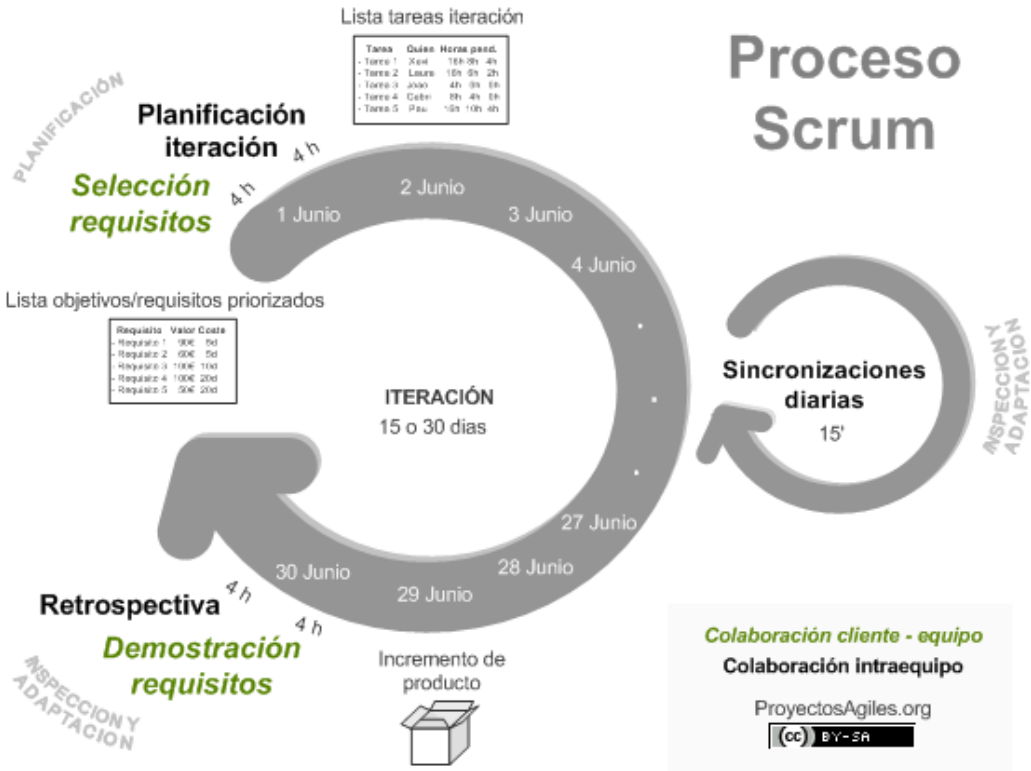

Figure 1: Proceso SCRUM `http://www.proyectosagiles.org/`

Se define el Product Backlog o pila de producto, y la descripción y ejecución de cada uno de los sprints. La plantilla propuesta contiene toda la información detallada en cada una de sus secciones y es presentada en la tabla nro. 1.

5 Implementación de la Propuesta

Para la implementación de esta propuesta se han considerado cuatro grupos de estudiantes en condiciones académicas de iniciar su trabajo final de la carrera Analista en Computación. En primer lugar, se fijó el inicio del proyecto en agosto del año 2013 con un plazo de finalización para finales del mes de febrero de 2014. Los cuatro equipos de trabajo estuvieron conformados por tres estudiantes. Se tuvo especial cuidado en la selección de los proyectos a desarrollar. Todos ellos han tenido características similares en cuanto a tamaño, nivel de complejidad y tipo de proyecto. En todos los casos se eligió un sistema de gestión comercial.
Se mantuvo una reunión general con los directores de los proyectos y los estudiantes para presentar la plantilla, fijar claramente el alcance de cada proyecto y acordar la metodología de trabajo en general. Luego se realizaron reuniones mensuales que permitieron determinar el seguimiento y utilidad de la plantilla, es decir si ordenaba el trabajo y si les permitía descubrir si la planificación prevista se ajustaba a la realidad. De no ser así intensificar las acciones necesarias para limitar las desviaciones producidas.

Tabla nro.1. Plantilla para documentar proyectos de finales

<CARÁTULA> Título del Proyecto Integrantes del grupo Director del Proyecto Carrera - Año - Facultad - Universidad
Índice **1. Introducción general** Explicar brevemente el propósito del proyecto, a quien está dirigido, el contexto, el objetivo principal y los objetivos secundarios.
2. Captura de Requisitos **2.1 Definición del SRS(Software Requirements Specification)** Definición del SRS. Para cada funcionalidad del SRS debe poseer una descripción del flujo de eventos modelado con un diagrama de actividades UML o una plantilla para la descripción de cada funcionalidad. Definir los límites del proyecto. **2.2 Modelo de Contexto** El modelo de contexto determinará donde estará inmerso el sistema. También define una arquitectura para el sistema. Para este punto se requiere la confección de un diagrama de clases UML.
3. Definición del artefacto: Product Backlog Listado de todas las funcionalidades que contendrá el sistema.
4. Definición y planificación de los Sprints. **4.1 Definición de un diagrama de Gantt con la planificación de los Sprints.** **4.2 Definición del Sprint Backlog o pila del sprint** Repetir para cada sprint. Documentar el incremento a lograr al finalizar el sprint (funcionalidades que tendrá el sistema al finalizar el sprint - el entregable)

5. Ejecución de los Sprint Para cada Sprint se producirá un documento que contenga: **Sprint** <id sprint> **Baclock id** <Id> **Historia** <nombre> **Responsable**<nombre del responsable> . **Descripción de la funcionalidad** Propuesta para dar solución al problema: Diagrama de clases de diseño junto con explicación de las decisiones tomadas. Implementación de la funcionalidad, detalles de la/las tecnología/as utilizada/as: (es posible registrar alguna cuestión técnica particular, por ej. la utilización de librerías externas o módulos específicos). Diagrama de secuencia UML con escenario **Prueba**. Definición de las pruebas para la tarea. Artefacto Gráfico de avance **(Burn Down)**. Gráfico que muestra el estado de avance del trabajo del sprint en curso. <se repite para cada funcionalidad de cada sprint>
6. Retrospectiva del Sprint (Sprint Retrospective). Al finalizar cada sprint, se lleva a cabo una retrospectiva del sprint (sprint retrospective), en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint. El propósito de la retrospectiva es realizar una mejora continua del proceso. Considerando ¿Qué ha ido bien en el sprint? ¿Qué podría mejorarse en el siguiente sprint? Se producirá una descripción textual con los aportes de los miembros del grupo de trabajo.
7. Conclusiones Incluir las conclusiones respecto del proyecto, lo realizado, el impacto del proyecto y el valor agregado que brinda su implantación. También es posible incluir lo que queda por realizar y se propone dejarlo para una siguiente etapa por estar fuera de los límites establecidos al comienzo del proyecto.
Bibliografía consultada Incluir manuales, páginas web o cualquier otro medio de donde se haya obtenido información para el desarrollo del proyecto.
Anexos – Técnicas de Modelado usadas Resumen de las técnicas y lenguajes de modelado utilizados. Describir las herramientas de software usadas en el proyecto. Podrá incluirse algún otro anexo en caso de ser necesario.

6 Conclusiones

Este trabajo surge de un Proyecto de Investigación e Innovación para el Mejoramiento de la Enseñanza de Grado (PIIMEG) en el que se involucran asignaturas del último año de la carrera Analista en Computación y del tercer año de la carrera Licenciatura en Ciencias de la Computación. En estas asignaturas se abordan diferentes metodologías de desarrollo de software, técnicas de estructura y comportamiento de bases de datos y todos los temas asociados a la gestión de proyectos de software. El Proyecto final, es donde los estudiantes desarrollan un proyecto de software real, aplicando todos los conocimientos adquiridos durante la carrera. El abordaje consiste en investigar y detectar las causas que hacen que, en un importante número, los estudiantes no logren cumplir con la planificación establecida para desarrollar el Proyecto final de carrera, aún cuando la misma se va ajustando durante la realización, se cuenta con todas las herramientas metodológicas y de aprendizaje, con un cuerpo de docentes con años de experiencia en el dictado de estas asignaturas y con un seguimiento personal de cada grupo de estudiantes que realiza el proyecto. Y si bien se conocen algunas causas como son la falta de organización por parte de los estudiantes, dificultades para el trabajo en equipo y demasiada carga horaria, los autores de este tra-

bajo también se ocuparon de estudiar, detectar y analizar qué sucede y cómo se realizan las estimaciones, la gestión de riesgos, la planificación temporal, la gestión de configuración del software y los ajustes durante el avance del proyecto. Además, se conoce que la problemática abordada, se da tanto en los proyectos finales de carreras de computación como en la industria del software en general. En el marco del proyecto PIIMEG se analizaron datos de los proyectos finales realizados en los últimos cinco años, y se relevaron fechas de inicio y fin, cantidad de participantes, tamaño del proyecto y actividades de gestión del proyecto realizadas; luego se hizo un análisis comparativo entre dichos proyectos, se obtuvieron y analizaron las conclusiones y se propusieron acciones correctivas, que luego de aplicarlas se midieron nuevamente los resultados obtenidos. Se refinaron el diseño de las encuestas considerando nuevas variables o modificando las existentes, sobre la base de la experiencia adquirida luego de haber tomado las encuestas una vez. Una acción correctiva propuesta e implementada durante el primer cuatrimestre de 2011 y 2012, y también se está desarrollando en la actualidad, fue proponer un único proyecto integrador entre las materias de 3er año, para que de esta forma los alumnos se vieran beneficiados principalmente en la integración de los conocimientos y optimizando los tiempos de desarrollo de los proyectos. Este taller integrador fue un espacio de interacción generado entre las asignaturas involucradas, con la participación de docentes y estudiantes. Este espacio, resultó un ámbito sumamente enriquecedor para el intercambio de ideas, planificación de actividades y evaluación de procesos. La principal ventaja que resultó de la implementación de esta medida, es que se pudo minimizar la carga horaria en los estudiantes, ya que no deben elaborar un proyecto diferente por cada asignatura. En este trabajo se propone otra acción correctiva concreta tendiente a guiar, ayudar y organizar al estudiante el desarrollo de un proyecto de software como proyecto final de carrera, que consiste en una plantilla, basada en SCRUM, que guía a cualquier desarrollador de un producto de software pequeño y de baja o mediana complejidad, y en particular, sirve para contextualizar el trabajo final de los estudiantes de las carreras de computación que deben desarrollar una aplicación de software como resultado de su proyecto final. De los cuatro equipos de estudiantes que implementaron esta propuesta, es decir, utilizaron la plantilla para documentar y desarrollar su trabajo final de la carrera Analista en Computación, solo uno de ellos no logró culminar su trabajo en los tiempos previstos, y lo hará con un retraso de 45 días. La principal causa de este retraso se debió a un inconveniente que tuvo el equipo de estudiantes que no pudo definir y dividir claramente sus roles de trabajo, y además, uno de los integrantes comenzó a trabajar y descuidó las tareas que le fueron asignadas. Por otra parte, cuando en las reuniones de seguimiento se detectaron desviaciones del equipo en la planificación temporal, se pudo probar que esta plantilla les permitió contar con una herramienta de autogestión y autocontrol facilitándoles reorganización sin demorar la finalización de su proyecto final. Si bien tres de los cuatro grupos de trabajo elegidos para implementar esta propuesta terminaron con éxito su proyecto final en los tiempos previstos, y manifestaron en la encuesta están muy conformes con la propuesta, el desafío es continuar trabajando con otros equipos de estudiantes, y también realizar comparaciones con estudiantes que desarrollan el trabajo final sin utilizar esta plantilla aquí propuesta.

References

[1] Tom DeMarco. Libro: Structured Analysis and System Specification, 1979.

[2] Pressman, Roger. Software Engineering: A Practitioner's Approach. 7ma edition. McGraw Hill. 2006.

[3] Pankaj Jalote. An Integrated Approach to Software Engineering. Springer 2006.

[4] Kent Beck, Mike Beedle y otros. Agile Manifiesto and agile principles. www.agilemanifiesto.org. Febrero 2001.

[5] Scrum in Action: Agile Software Project Management and Development. Andrew Pham, Phuong Van Pham. Course Technology Ptr. 2011.

[6] Marcela Daniele. Nicolás Florio. Daniel Romero. Definición y uso de Plantillas Genéricas para la descripción de Casos de Uso. Proyecto de Innovación e Investigación para el Mejoramiento de la Enseñanza (PIIMEG 2004), Secretaría de Ciencia y Técnica y Secretaría Académica, Universidad Nacional de Río Cuarto. RR Nº 302/2004.

[7] Marcela Daniele. Daniel Romero. Evolución de Plantillas Genéricas para la descripción de Casos de Uso a Plantillas Genéricas para

Análisis y Diseño. Proyecto de Innovación e Investigación para el Mejoramiento de la Enseñanza (PIIMEG 2006), Secretaría de Ciencia y Técnica y Secretaría Académica, Universidad Nacional de Río Cuarto. RR Nº 109/2005.

[8] Marcela Daniele. Daniel Romero. La enseñanza de gestión de proyectos de software y la aplicación de herramientas que favorezcan su automatización. Secretaría Académica y de Ciencia y Técnica, Universidad Nacional de Río Cuarto. RR Nº 499/06. (01/08/2006, 31/07/2008).

[9] Marcela Daniele, Fabio Zorzan, Paola Martellotto, Mariana Frutos, Marcelo Uva, Ariel Arsaute, F. Brusatti, J. Guazzone, S. Angeli. Estimación y Planificación de Proyectos de Software versus duración de proyectos finales en la carrera Analista en Computación. Secretarías Académica y de Ciencia y Técnica, Universidad Nacional de Río Cuarto. RR Nº /11. (01/02/2011, 31/12/2012).

[10] Marcela Daniele, Fabio Zorzan, Paola Martellotto, Marcelo Uva, Ariel Arsaute, Mariana Frutos. Causas que producen que los estudiantes de Computación retrasen la culminación de su Trabajo Final. Secretarías Académica y de Ciencia y Técnica, Universidad Nacional de Río Cuarto. Presentado en abril de 2013. En evaluación.

[11] Fabio Zorzan, Mariana Frutos, Ariel Arsaute, Marcela Daniele, Paola Martellotto, Marcelo Uva, Carlos Luna Delayed Completion of Final Project of the Career Computer Analyst: Seeking its Causes. XX Congreso Iberoamericano de Educación Superior (CIESC 2012), en el Marco de la XXXVIII Conferencia Latinoamericana en Informática – CLEI 2012 - Octubre 1 al 5 de 2012 - Medellín, Colombia. ISBN 978-1-4673-0792-5.

[12] Edward Yourdon. Análisis Estructurado Moderno, 1ra Edición. Yourdon Press, 1989.

[13] Bertrand Meyer. Design by Contract. Publisher Prentice Hall PTR, 2000.

[14] Ivar Jacobson, Grady Booch, James Rumbaugh. El proceso unificado de desarrollo de software. The Addison-Wesley Object Technology Series. Félix Varela, 2004.

[15] James Rumbaugh, Ivar Jacobson, Grady Booch. El Lenguaje Unificado de Modelado: Manual de Referencia.Addison-Wesley object technology Addison Wesley, 1999.